Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

Architecture for Enhanced Description, Discovery and Exchange of Services in P2P Networks

Ramesh Subramanian
Quinnipiac University

Brian Goodman
IBM Corporation

Architecture for Enhanced Description, Discovery and Exchange of Services in P2P Networks

Ramesh Subramanian
Quinnipiac University – School of Business
Ramesh.Subramanian@Quinnipiac.edu

Brian D. Goodman
IBM Corporation
bgoodman@us.ibm.com

ABSTRACT

Over the last few years, peer-to-peer (P2P) computing has emerged as one of the more promising and exciting new technologies in the field of Information Technology. This reflects the general shift from the traditional client/server model of computing to peer-to-peer computing. A "peer" on a (P2P) network is simply a client and a server on the same footprint. One could argue that all things on a network are peers ─ they merely speak different languages and perform different functions. This paper addresses two specific problems that arise in P2P implementations, namely: how can peers describe themselves, and how can a peer discover and exchange services with other peers ─ using existing standards. We have developed an architecture that merges P2P computing with web services technologies which offers a viable solution to the problem and will accelerate the adoption of P2P computing within organizations.

KEYWORDS

P2P computing, distributed computing, peer description, peer services, peer discovery, web services, WSDL, UDDI.

INTRODUCTION

Peer-to-peer (or "P2P") computing has aggressively moved to the center-stage of the computing field in recent years. P2P computing emphasizes the shift away from centralized and client/server models of computing to a fully decentralized, distributed model of computing and content distribution. P2P has been described as "a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet" (Shirky, 2000). We can also add "processes and applications" to this list. In the P2P mode of thinking, a "peer" system can be almost any computing device connected to a network – whether it is a computer, a printer, a facsimile machine, a video camera or an email server. This idea has great ramifications in the way people work, learn and share resources in organizations.

The importance of P2P computing in organizations has been made apparent in several recent research reports. According to a Gartner Consulting Report (Gartner Consulting Report, 2001) published in 2001, there exists a very high probability that "half of the current server-based content management vendors will add Data Centered P2P functionality to their product offerings by 2005."

Smith, Clippinger and Konsynski (Smith et al., 2003) state that P2P computing represents a major technology shift and suggest that corporations should seriously start focusing on P2P computing, "whether they like it or not."

Three areas where P2P computing offers promise are data management (including distribution), data-processing and service-processing.

Data Management

The emphasis is shifting from storing data in, and serving data from, centralized servers to storing and serving (at least some of) the data/content from the client-side. The content provider thus manages his/her content in a local client, and shares them with anyone who is authorized to access them. The responsibility for content creation, storage and security dwells on the client side. There are several advantages to this approach. By shifting the responsibility for content to the client side, server-side management of diverse resources can be vastly reduced. Server managers need not be responsible for the integrity of the content. Problems arising from centralized distribution of content could be averted. The disadvantages of this approach include factors such as reduced security and reduced integrity of content arising from client-side mismanagement or misconfiguration.

Data Processing

P2P distributed computing has increasingly become a popular and successful method for processing massive quantities of data on a wide variety of system architectures and configurations. Two notable examples are distributed.net (The Distributed.net website) and SETI@Home (The SETI@HOME website). Both projects rely upon extraneous users to run proprietary client software to perform data analysis on network-obtained work units. Millions of users donate the extra CPU cycles of their machines to specific causes. These systems have proven the viability of leveraging the abundance of idle CPU-cycles of computers on the Internet. One shortcoming, however, is that there is no way yet to access these CPU cycles using more generic, non-proprietary methods. Furthermore, these have no ability to identify the true client-environments in which their software is running, thus preventing further optimization/cost analysis. Most of these projects are happy enough to get the work units processed by peers, and there are so many computers working on the same project that statistically a few slower machines are not a serious problem.

Service Processing

The concept of peers providing various applications and services to other peers *on demand* is gaining traction, especially through the technology of web services. Web services are applications and services provided by "service providers" that developers and/or other applications can interact with or invoke over the network. Web services are essentially remote procedure calls, and are thought of as a server-based architecture. In comparing web services with P2P computing, we notice that peers on a P2P network are simply clients and servers on one footprint. Thus, one could argue that all things in and on a network are peers – they simply speak different languages and perform different functions ─ *and* that they are also servers providing various services. Thus, web service providers are also a type of peer. P2P service processing combines P2P and web services technologies.

Currently there is increased interest in exploiting P2P computing within organizations in all of the above areas. Our focus in this paper in on P2P service processing. However, before P2P computing can be used for service processing, two important issues require to be resolved which affect the discovery and use of P2P services. They are:

1. *How does a peer describe its wares, and*
2. *How does a peer advertise its services*

In this paper we address these two issues with a view to enhancing P2P Service discovery and processing. We describe an open-protocol/open-architecture solution for enhanced description, discovery and exchange of services in P2P distributed networks. We draw ideas from current approaches to P2P computing as well as web services technologies such as UDDI, WSDL and SOAP. We believe that our approach offers greater flexibility and an easy-to-use method for developing and deploying a diverse range of P2P distributed computing applications.

The paper is organized as follows: we first set the context to what follows by discussing P2P computing – its history and some recent trends ─ keeping in mind the two problem areas that we listed above. In the following section we describe the problems ─ i.e., the description, discovery and exchange of services in current P2P distributed networks in greater detail. In the next section we describe in detail our solution to the problems and discuss how our approach differs from other works in this area. In the final section, we discuss the current status of our research and our plans for the future.

P2P AND DISTRIBUTED COMPUTING: BRIEF REVIEW OF CURRENT SOLUTIONS

Several P2P implementations have been proposed in recent years. Most of the P2P applications are *file-centric*, and facilitate data discovery and either synchronous or asynchronous file sharing (e.g. Napster, Gnutella (Kan, 2001), KaZaa). A few applications also provide access to resources other than files i.e. the SETI@Home project, which enables CPU-cycle sharing. The notion of "peer-to-peer" computing systems is, however, hardly new. In the 1960s, the Internet was conceived as a peer-to-peer system, and the first few nodes on the ARPANET (UCLA, SRI, UCSB and University of Utah) were connected together as equal computing peers (Minar and Hedlund, 2001). Other examples of long established P2P systems include the Usenet and DNS. Usenet, which has been around since 1979, is a system that copies files between computers using no central computer. The DNS (Domain Name System) can also be considered to be a P2P system in that each host can either function as a server or client, propagating data requests (i.e., a domain name query) across the network.

Other systems that share similarities with P2P systems include distributed networked systems such as FTP (File Transfer Protocol) and Telnet, CORBA (Common Object Request Broker Architecture) and DCOM™(Distributed COM), an extension of Microsoft's COM (Common Object Model) (Gisolfi, 2001) and Sun Microsystems' JXTA (Gong, 2001). Of these, JXTA is relatively new. The JXTA framework offers a set of six protocols (Peer Discovery Protocol, Peer Resolver Protocol, Peer Information Protocol, Peer Membership Protocol, Pipe Binding Protocol, and Endpoint Routing Protocol). JXTA is transport-independent and can utilize TCP/IP as well as other transport standards. It is meant to be a conceptual framework that provides some protocols and mechanisms using which one can implement either a centralized or decentralized P2P system. In our opinion, JXTA offers a robust and flexible P2P solution framework. However, it must be noted that the protocols it offers are not standard.

In fact, the reality is that none of the above technologies are entirely open. Therefore, solutions that are built using these technologies are typically dependent on a specific vendor's implementation.

PROBLEMS IN DESCRIPTION, DISCOVERY AND EXCHANGE OF SERVICES IN P2P DISTRIBUTED NETWORKS

There are two main problems: the peer self-description problem and the peer self-advertisement problem.

The peer self-description problem (or "how does a peer describe its wares?")

One can say that peers are much like web services. To understand the similarities of the two, we provide a short description of web services here.

Web-services are based on a server-centric environment. In this environment, the web services reside on the servers, and provide descriptions of themselves on the central servers. The services advertise functions they perform explicitly as there is no implicit understanding of what they do. However, the shortcoming is that the advertisement is server-based and is static. The server is often unrelated the implementation of the service. If a service "dies" or is removed from a provider, that information is not immediately known to the central server. Web services comprises of these key technologies:

- Simple Object Access Protocol (SOAP) (W3C.org, 2000): SOAP uses an XML formatted message to communicate between applications. However, we contend that the development of the SOAP protocol is not necessarily appropriate as a method by which electronic circuits or peers should communicate.
- Web Services Description Language (WSDL) (W3C.org, 2001): WSDL is a method by which web based services describe themselves to developers. This methodology was clearly the inspiration for this research.
- Universal Description, Discovery and Integration (UDDI) (UDDI.org, 2000): UDDI is a platform-independent open framework for advertising described services. It affords search-based discovery and aids in the integration of services into other business processes. It is the phone book for services.
- Remote Procedure Call Router (RPC Router): Remote Procedure Call Router is for identifying the requesting method and dynamically invoking code to respond to that method invocation.

Currently web services have limited self-description ability. "Self-description" is the ability of a piece of software code (i.e., a web service) to self-document in such a way that another programmer can understand how to interact with and use that web service. For example, the service could highlight the arguments (inputs) it expects to receive, and the return types (output types) it will return upon execution. Currently services fail to support a rich enough level of description of the inputs and outputs that would enable a non-human (such as a computer program) to use the services effectively. The accessor of the service might understand that certain services are available, but a service with the name "MyCalculator" won't mean anything to a computer program, even though a human looking at the service could easily infer that the service provides certain mathematical operations.

Peers (in a P2P context), suffer from the same problem as web services -- even with current service description standards, peers are unable to describe their wares completely enough.

The peer self-advertisement problem (or "how does a peer advertise its services?")

There are two popular P2P architectures in use today, "true" P2P, and P2P with a central server. "True" P2P infrastructures use user intervention to learn about other peers in the network, and then join the network. Each peer then announces itself to

other peers in the network. In a "server-centric" P2P network, individual peers connect to the network through a main or distributed hub (i.e. server). The server then announces the peer to the other peers.

Current P2P implementations are centered on providing resource-sharing functionality, and this is understood to be the implicit function of a peer.

The above discussion leads to the following conclusion: currently there is no generic way for peers, either within the P2P context or the Web services context, to describe themselves, or advertise the exact services they offer. It is our belief that web services technologies can be adapted and merged with P2P technologies in order to solve the problems in description, discovery and exchange of services in P2P distributed networks.

Our solution

Our solution to the problems described above is as follows:

- Provide the means for peers to **describe themselves** more clearly in the networked environment, and
- Provide the means for peers to **advertise their services** in an enhanced manner

Better description capability

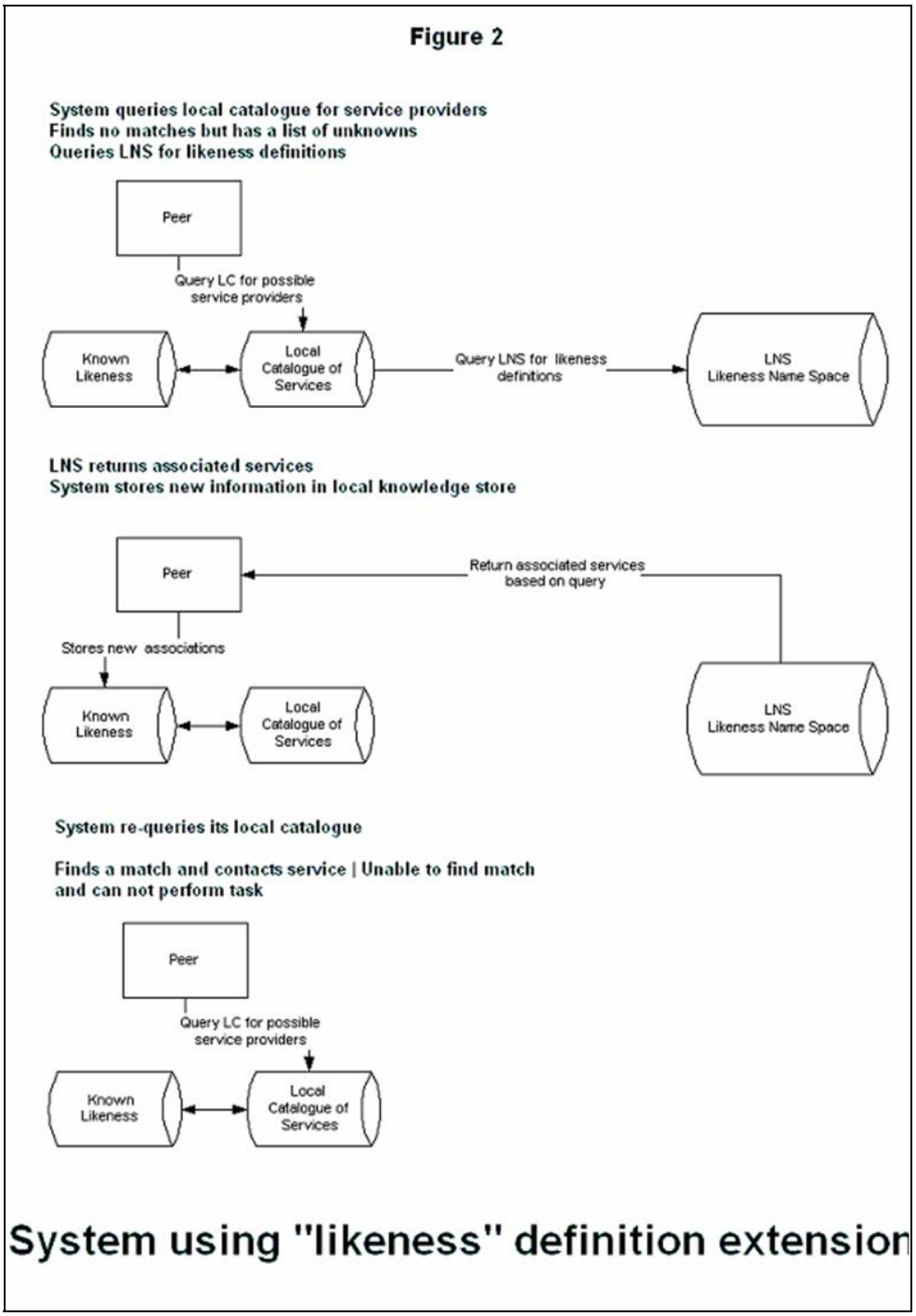
Every peer will be empowered to describe its API (i.e., its Application Programming Interface – the means through which an application "exposes" itself to other application) in self-describing markup. An example of current markup technologies would be WSDL. It would be acceptable to extend this standard but it is not a necessary requirement. Currently data types are specified by name space/standards. In addition to these publicly-used identifiers we introduce a new component: likeness. Instead of simply specifying a data input as a string, it can be specified as being like a password. In the same way, services can also express their likeness with relation to other services. Furthermore methods contained by these services can also inherit the ability to express likeness. Thus, at every level (service, method, argument, return-type) a namespace will specify each element's likeness – i.e., what is it like. A base set of standard components can be defined and services can point to each other as usage grows. An illustrative example of using WSDL to self-describe a peer and enhancing it with the notion of likeness is given in figure 1. Simply put, WSDLs describe the interface for a web service. By adding a new namespace to the document (a standard practice) we enable the referencing of elements defined elsewhere. In this case, likeness elements, are considered the basic building blocks. The idea is to make a parameter such as URL defined as type "String" more descriptive. A machine doesn't know what type of "String" this is. It does not know what a URL is, URL is just a unique identifier - could just as easily be "FOO". By adding a likeness attribute we can point to a definition in the new namespace referenced at the beginning of the document which defines a URL, probably in terms of an object (shown in Figure 1 under **<message name="getInput">)**. Figure 1 shows a peer whose name is defined as a "Generic Service." Towards the center of the figure can be seen an operation "get" which is defined as having a likeness to the HTTP operation "get" (<operation name="get" likeness="**xsdextended:http:get**">.

Figure 1

```
<?xml version="1.0"?>

<definitions name="GenericService"
             targetNamespace="urn:com-ibm-webahead-services-genericservice"
             xmlns:tns="urn:com-ibm-webahead-services-genericservice"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/1999/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsdextended="http://www.w3.org/1999/ExtendedXMLSchema">

   <message name="getInput">
      <part name="url" type="xsd:string" likeness="xsdextended:url"/>
   </message>

   <message name="getOutput">
      <part name="stream" type="xsd:string" likeness="xsdextended:html"/>
   </message>

   <message name="connectionInput">
      <part name="type" type="xsd:integer"/>
      <part name="url" type="xsd:string"/>
   </message>

   <message name="connectionOutput">
      <part name="stream" type="xsd:string"/>
   </message>

   <portType name="GenericService">
      <operation name="get" likeness="xsdextended:http:get">
         <input message="getInput"/>
         <output message="getOutput"/>
      </operation>
   </portType>

   <binding name="GenericServiceSoapBinding" type="GenericService">
      <soap:binding style="rpc"
                    transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="get">
         <soap:operation soapAction=""/>
         <input>
            <soap:body use="encoded"
                       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="urn:com-ibm-webahead-services-genericservice"/>
         </input>
         <output>
            <soap:body use="encoded"
                       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="urn:com-ibm-webahead-services-genericservice"/>
         </output>
      </operation>
   </binding>

   <service name="GenericService">
      <port name="GenericSerivcePort" binding="GenericServiceSoapBinding">
        <soap:address location="http://suds.adtech.internet.ibm.com/soap/rpcrouter"/>
      </port>
   </service>

</definitions>
```

Example of extending self-describing mark-up

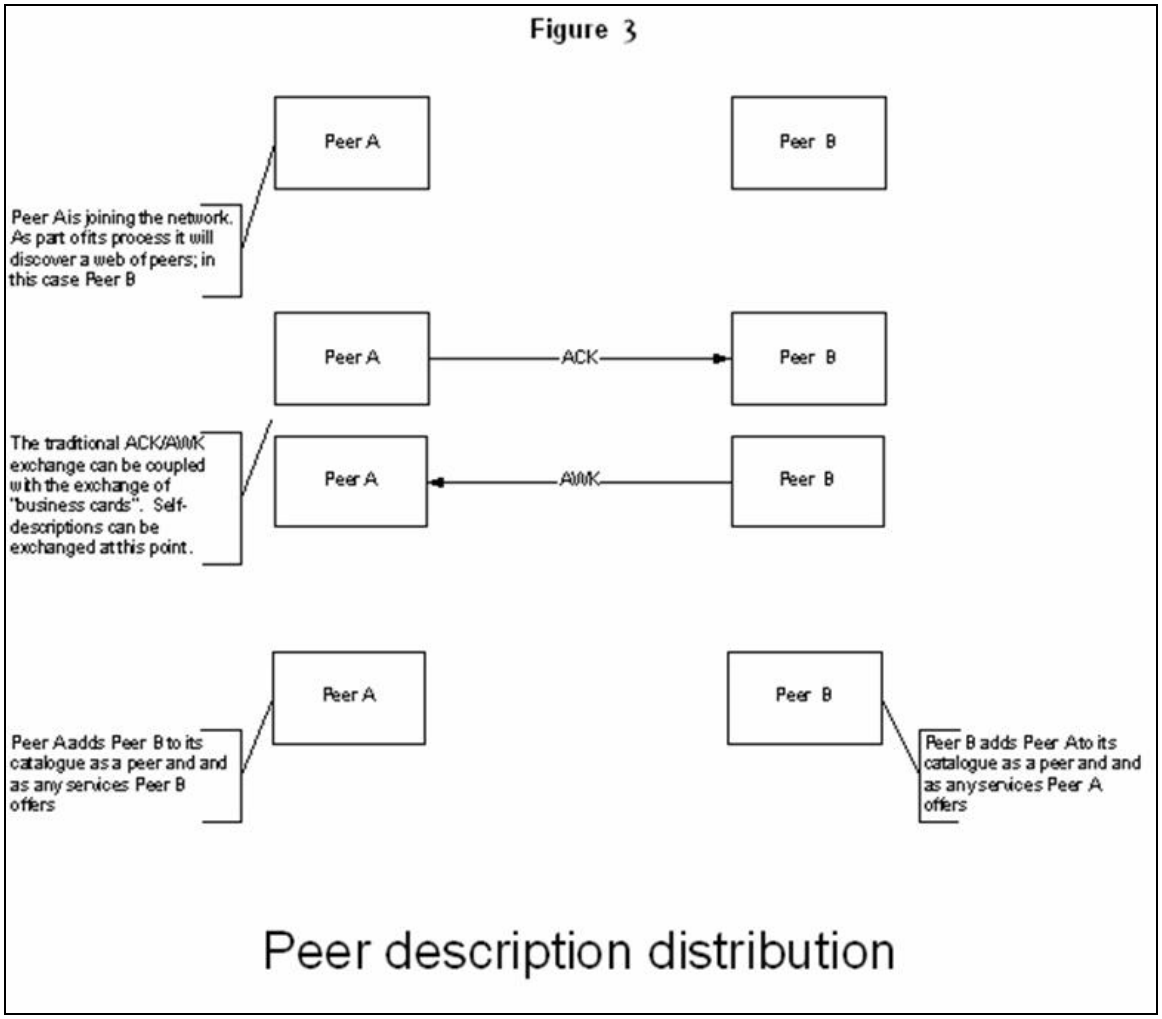
How can the peer "likeness" concept be used in practice? We propose a **central store** that can house an association-list of possible likeness of peers. Currently we have identified this server as the **Likeness Name Space (LNS)**. How does this server work? Provided a stated likeness, the server will return a list of associated elements. It provides a system for computer code to locate a service without previous knowledge of the likeness token. A system using the peer likeness extension is illustrated in figure 2. In the figure, a hypothetical peer system queries a local catalogue for particular service providers. Upon not finding any, it queries the LNS for "likeness" definitions. The LNS returns a set of "like" services, which the peer saves in its catalog. Future searches are made using this catalog.



Peer description distribution

Each peer in a distributed network distributes its self-describing markup (described above) as a part of a handshake process. The business card of sorts is then stored in a local catalogue of known services by each peer involved in the handshake. Thus a query to the network for known services is dynamic and timely. Current technologies such as UDDI are based on a master-slave architecture where the directory/catalogue is stored server side. The catalogue has no direct connection to the services in the network. Moving this functionality to the peer allows for real time resource discovery and integration. Cataloguing the community of peers in this manner also provides redundancy. A peer with a catalogue that has become out of date can

find alternative peer-services to perform the desired function. An optional query to the network may afford specifying the resource desired, allowing only the capable services to reply. Another benefit is that when certain services are no longer offered by a peer, instead of re-cataloging part of the network, a broadcast message can provide an immediate listing of appropriate services. Figure 3 illustrates how the Peer description distribution takes place between two peers, A and B. Figure 3 describes how a new peer (A) joining a network discovers another peer (B) and exchanges "business cards," which are self-descriptions. After the exchange, peers A and B then add B and A respectively to their local catalogs.



The operation of this system becomes apparent by using an illustrative example.

Example: A peer might want to send some e- mail. It looks through its local directory for SMTP. It doesn't find any appropriate services; however, it is left with a few services it does not know about, for instance, mail and instant messaging. It takes both of these likeness tokens and queries the LNS. The Likeness Names Space server returns a list of elements for each. The peer adds the definitions to its local knowledgebase and re-queries its local directory for a possible SMTP service. In this case one of the elements for SMTP was Internet Mail. The peer now knows Internet Mail can also mean SMTP and that directory entry can be of assistance. This is illustrated in figure 4, and is self explanatory.

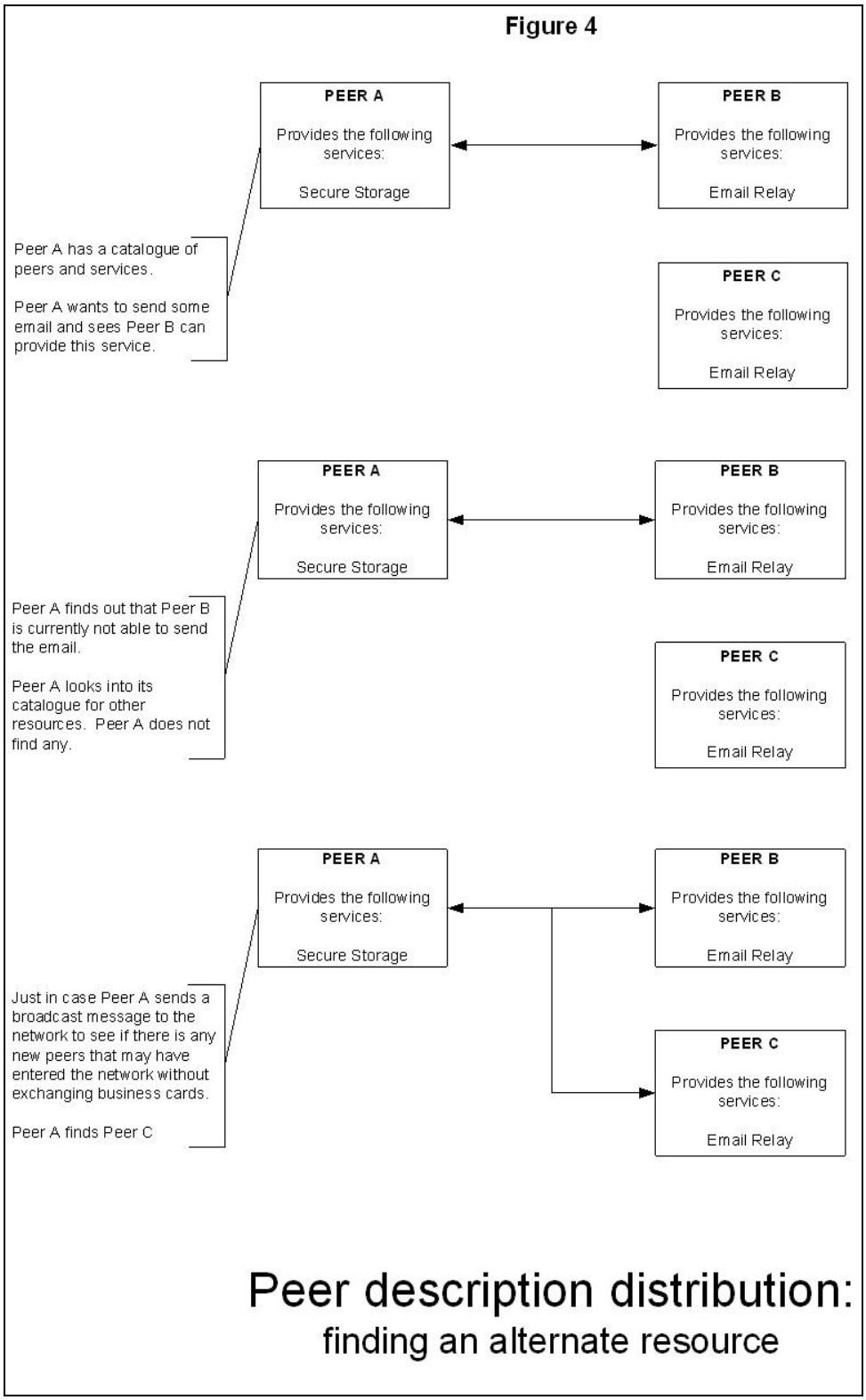


Figure 4
PEER A
Provides the following services:
Secure Storage
PEER B
Provides the following services:
Email Relay
PEER C
Provides the following services:
Email Relay
Peer A has a catalogue of peers and services.
Peer A wants to send some email and sees Peer B can provide this service.
PEER A
Provides the following services:
Secure Storage
PEER B
Provides the following services:
Email Relay
PEER C
Provides the following services:
Email Relay
Peer A finds out that Peer B is currently not able to send the email.
Peer A looks into its catalogue for other resources. Peer A does not find any.
PEER A
Provides the following services:
Secure Storage
PEER B
Provides the following services:
Email Relay
PEER C
Provides the following services:
Email Relay
Just in case Peer A sends a broadcast message to the network to see if there is any new peers that may have entered the network without exchanging business cards.
Peer A finds Peer C
Peer description distribution:
finding an alternate resource

Advantages of our solution

The solution that we offer is a clear enhancement to the current art: Current self-describing markup standards, such as WSDL, offer a limited form of description. Other proprietary protocols know the services implicitly; description is limited only to the protocol. *Expanding self-describing markup to include a likeness or categorization attribute allows for data types, methods, and services to be known more than just predefined data types.* Moreover being able to categorize the service itself allows for peers to generalize the service and act on it without knowing the very specific differences. Currently rich descriptions in markup languages allow developers to understand the markup. However, our proposed solution would allow machines, such as applications, to decode the markup as well. The specification of likeness allows for a richer definition of properties, process and peers."

Current service publishing architectures, such as UDDI, are detached from the services. The current art has almost no ability to aid in offering real time services. It is a static resource. The proposed solution places the descriptions in the distributed network. Having the peers express their current API allows all peers to have reliable connections to services. Moreover, this affords the peers real time service discovery and autonomy from central server architecture.

CONCLUSIONS AND NEXT STEPS

We developed the idea that the P2P and web services technologies could be combined to create a distributed service sharing system in which each peer could describe itself to other peers in the form of a "likeness" description. We then developed a framework which uses only open source technologies in order to architect such a system. Our system provides a way for the peers to advertise themselves to the network. We feel that our architecture is fundamentally good and can be used to solve the problem of peers not being able to describe themselves adequately to other peers.

This is an idea that is gaining traction in the literature. For example, Alfred Loo (2003) suggests the concept of combining web services with P2P computing to enable CPU power sharing, which is similar to our own concept. The main difference is that Loo suggests the use of a Java Application Program (a 'power server') that will act as the broker between various peers (i.e. the 'need' peers and the 'want' peers). The major enhancement proposed in Loo's paper is the use of a 'coordinator' that allows new users to register, maintain a database of power servers, matches user requirements with power servers and transfers servlets to power servers. In our model, we use an even more pure form of P2P service sharing though the definition of a mechanism to describe and advertise the services that each peer requires or provides in a completely P2P manner, without using a central broker.

The next step in our research is to implement this system and test its performance in the real world. This step can be undertaken in stages. The *first stage* could just deal with the problem of data distribution, and the *second stage* could involve data and service description and advertisement.

For the first stage, we have already developed a basic P2P resource description and resource sharing system that allows enhanced description through the incorporation of metadata facilities (see Subramanian and Goodman, 2003). This system will be enhanced to include the notion of business-card catalogues. After that is implemented, we plan to initiate testing it to ascertain proof of concept.

In conclusion, we believe that this system, if successful, will herald the next advancement in intra-and inter-organizational resource-sharing by providing enhanced description, discovery, and exchange of services in Peer-to-Peer distributed networks.

REFERENCES

1. Gartner Consulting Report (2001), "The Emergence of Distributed Content Management and Peer-to-Peer Content Networks," *GartnerGroup Report # 010022501,* 2001.
2. Gene, K. (2001.) "Gnutella," in *Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology*, pp 94-122, Edited by Andy Oram, O'Reilly & Associates, CA, USA.
3. Gisolfi, D. (2001) "Web services architect, Part 3: Is Web services the reincarnation of CORBA?" available online at http://www-106.ibm.com/developerworks/webservices/library/ws-arc3/resources. (2001). Last accessed May 6, 2003.
4. Gong, L. (2001) "JXTA: A Network Programming Environment," IEEE Internet Computing, Vol. 5, No. 3, May/June 2001, available online at http://computer.org/internet/v5n3/w3jxta.htm#f1.
5. Loo, Alfred W. (2003) "The future of peer-to-peer computing," *Communication of the ACM*, September 2003, Volume 46, No. 9.

6. Minar, N. and Hedlund, M. (2001) "A Network of Peers: Peer-to-Peer Models through the History of the Internet," in *Peer-to-Peer: Harnessing the power of disruptive technologies*, Edited by Andy Oram, O'Reilly & Associates, CA, USA. 2001. Pp3-20.
7. Shirky, C. (2000) "What is P2P...And What Isn't,**"** (November 2000). Available online at http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html. Last accessed May 6, 2003
8. Smith, H.A., Clippinger, J., Konsynski, B. (2003) "Riding The Wave: Discovering The Value Of P2P Technologies," *Communications of the AIS* (Volume 11, 2003) pp 94-107.The Distributed.net home page, http://www.distributed.net
9. Subramanian, R. and Goodman, B. (2004) "Peer-to-Peer Corporate Resource Sharing and Distribution with Mesh," in *e-Collaborations and Virtual Organizations,"* a compilation of peer-reviewed articles edited by Michelle Fong, published by The IDEA Group Inc., Hershey, PA. Forthcoming 2004. An initial version of the paper was published in *the Proceedings of the 14th IRMA International Conference, 2003.*
10. The distributed.net website, http://www.distributed.net
11. The SETI@Home website, http://setiathome.ssl.berkeley.edu/
12. UDDI.org, (2000) "UDDI Technical White Paper**,"** http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf.
13. W3C. (2000) "Simple Object Access Protocol (SOAP) 1.1," W3C Note 08 May 2000. http://www.w3.org/TR/SOAP/
14. W3C, (2001) "Web Services Description Language (WSDL) 1.1," W3C Note 15 March 2001. http://www.w3.org/TR/wsdl